Revista Ingeniería Biomédica
ISSN 1909-9762 / E-ISSN 1909-9991
Volumen 13 / Número 26 / julio-diciembre de 2019 / pp.37-45
Universidad EIA / Envigado, Colombia

Alineamiento gráfico de secuencias a través de programación paralela: un enfoque desde la era postgenómica

Johan S. Piña[1,ψ], Simon Orozco-Arias[2,3], Nicolás Tobón-Orozco[1], Mariana S. Candamil-Cortés[1], Reinel Tabares-Soto[4], Romain Guyot[4,5]

[1] *Equipo de investigación en bioinformática e inteligencia artificial, Universidad Autónoma de Manizales – Facultad de Ingeniería*

[2] *Departamento Ciencias Computacionales, Universidad Autónoma de Manizales Facultad de Ingeniería*

[3] *Departamento de Sistemas e Informática, Universidad de Caldas Facultad de Ingeniería*

[4] *Departamento de Electrónica y Automatización, Universidad Autónoma de Manizales Facultad de Ingeniería*

[5] *Institut de Recherche pour le Développement, CIRAD, University Montpellier, France*

Recibido 2 de febrero de 2019. Aceptado 29 de julio de 2019

Resumen— Un alineamiento gráfico o "dot plot" es un método de representación visual del análisis de datos genómicos, comúnmente utilizado para comparar la similitud de dos secuencias biológicas. El programa DOTTER desarrollado en 1995, es la herramienta más utilizada para este tipo de tareas. El mayor problema de este software radica en el elevado tiempo de ejecución para datos genómicos de gran escala. GEPARD (2007), realiza alineamientos más rápidos para secuencias más grandes que DOTTER, logrando reducir de esta forma el tiempo de ejecución del alineamiento de un cromosoma contra él mismo, de 382 años con DOTTER a 61 minutos con GEPARD, aunque con un nivel de detalle bajo debido a que utiliza un método de aproximaciones. En este artículo se propone una estrategia que trabaja sobre múltiples procesadores para realizar alineamientos a nivel genómico en menor tiempo de ejecución que GEPARD, logrando aceleraciones hasta de 27,9 veces utilizando 64 procesadores respecto al valor nominal. La estrategia permite la identificación de reorganizaciones cromosómicas, elementos repetitivos, comparación entre genomas de distintas especies y la medición de forma gráfica de la calidad de ensamblaje de secuencias genómicas rápidamente.

Palabras Clave— Alineamiento gráfico, Bioinformática, Computación de alto rendimiento, Programación paralela.

ψ Dirección para correspondencia: johan.pinad@autonoma.edu.co
DOI: https://doi.org/10.24050/19099762.n26.2019.1404

Graphical alignment of sequences through parallel programming: an approach from the post-genomic era

***Abstract*—**A graphical alignment or "dot plot" is a method of visual representation of genomic data analysis, commonly used to compare the similarity of two biological sequences. The DOTTER program, developed in 1995, is the most widely used tool for this type of task. The biggest problem with this software is the high runtime for large scale genomic data. GEPARD (2007), performs faster alignments for larger sequences than DOTTER, but reducing the execution time of the alignment of a chromosome against itself, from 382 years with DOTTER to 61 minutes with GEPARD, although with a low level of detail because it uses an approximation method. This article proposes a strategy that works on multiple processors to perform genomic-level alignments in a shorter run time than GEPARD, achieving accelerations up to 27.9 times using 64 processors from the nominal value. The strategy allows the identification of chromosomal rearrangements, repetitive elements, comparison between genomes of different species and the graphic measurement of the assembly quality of genomic sequences quickly.

***Keywords*—**Graphic alignment, Bioinformatics, High-performance computing, Parallel programming.

Alinhamento da sequência gráfica através da programação paralela: uma abordagem da era pós-genómica

***Resumo*—**Um alinhamento gráfico ou "dot plot" é um método de representação visual de análise de dados genómicos, comumente usado para comparar a semelhança de duas sequências biológicas. O programa DOTTER, desenvolvido em 1995, é a ferramenta mais amplamente utilizada para este tipo de tarefa. O maior problema com este software é o alto tempo de execução dos dados genómicos em grande escala. GEPARD (2007), realiza alinhamentos mais rápidos para sequências maiores do que o DOTTER, reduzindo assim o tempo de execução do alinhamento de um cromossoma contra si mesmo, de 382 anos com DOTTER para 61 minutos com GEPARD, embora com um baixo nível de detalhe porque utiliza um método de aproximação. Este artigo propõe uma estratégia que funciona em múltiplos processadores para realizar alinhamentos de nível genômico em um tempo de execução menor que o GEPARD, obtendo acelerações de até 27,9 vezes usando 64 processadores a partir do valor nominal. A estratégia permite a identificação de rearranjos cromossômicos, elementos repetitivos, comparação entre genomas de diferentes espécies e a medição gráfica da qualidade de montagem das seqüências genômicas rapidamente.

***Palavras-chaveo*—**Alinhamento gráfico, Bioinformática, Computação de alto desempenho, Programação paralela.

I. Introducción

Gracias a los avances de las tecnologías de secuenciación, se han incrementado en los últimos años la cantidad de información genómica disponible en las bases de datos de forma exponencial. A este periodo se le ha denominado la era de la Postgenómica [1]. La bioinformática, definida como la aplicación de técnicas computacionales para entender y organizar la información relacionada a las macromoléculas biológicas [2], [3] dentro de sus retos contempla el análisis de esta enorme cantidad de datos de forma eficiente, a través de las ciencias de la computación, las cuales aportan herramientas para analizar la información genética en sus diferentes especialidades [4]. Una de las tareas comunes en la bioinformática es el alineamiento de secuencias [5]–[7], que consiste en una forma de comparar dos o más cadenas de ADN o proteínas con el fin de mostrar las zonas de similitud para indicar relaciones entre genes de especies o individuos [8]. Existen varias formas de realizar alineamientos, su clasificación depende aspectos como el número de secuencias a alinear (pareado o múltiple), la región a alinear (local o global) o la forma en la que se presentan los resultados (plano o gráfico).

Concretamente los alineamientos gráficos son una técnica de alineación de secuencias que se encarga de comparar regiones individuales y globales de secuencias generando como resultado una imagen, denominada "dot plot". Esta imagen corresponde a un arreglo en dos dimensiones en el que se ubica una secuencia sobre el eje horizontal y la otra secuencia se dispone verticalmente. A continuación se asignan puntos que representan el grado de similitud de las secciones de las regiones que se están comparando [9].

La gráfica de puntos de secuencias de gran similitud se muestra como una única línea a lo largo de la diagonal principal de la matriz que compone la imagen. Los dot plots se utilizan para evaluar repetitividad en una sola secuencia graficando una secuencia contra ella misma, y las regiones que comparten similitudes significativas aparecerán como líneas fuera de la diagonal principal. Así mismo, en bioinformática se utilizan los dot plots para definir la calidad de un ensamblaje basado en referencia o para realizar análisis de secuencias en genómica comparativa y establecer relaciones genéticas entre individuos.

A lo largo de los años se han desarrollado diferentes softwares que permiten obtener alineamientos, tales como el algoritmo de Smith and Waterman [10] para alineamientos locales y el algoritmo de Needleman Wunsch para alineamientos globales [11]. Estos algoritmos utilizan los principios de programación dinámica para generar el puntaje de similitud óptimo, el cual se traduce en la intensidad del punto dibujado en el dot plot en los alineamientos gráficos. Otro método muy conocido para realizar comparaciones es a través de sufijos [12] que utiliza la heurística para definir el puntaje de alineamiento de las secuencias. Uno de los softwares tradicionalmente utilizados para la generación de dot plots es DOTTER (1995) [13] que utiliza la estrategia de Smith Waterman y posee una serie de herramientas que permiten modificar el resultado sin tener que recalcular el alineamiento; otro software comúnmente utilizado es GEPARD (2007) [14] el cual utiliza el método de sufijos para la comparación, logrando mejorar el consumo de tiempo y recursos computacionales, además cuenta con una interfaz gráfica para ingresar, visualizar y realizar anotación de la información.

Aunque estos programas son tradicionalmente utilizados para alineamientos gráficos, requieren una gran cantidad de recursos computacionales. La programación dinámica que utiliza DOTTER consume recursos de memoria RAM proporcionales al tamaño de las secuencias analizadas y utiliza un (1) procesador lo que dificulta el análisis de secuencias de gran tamaño. Por otro lado, GEPARD es más eficiente en tiempo debido a que utiliza una estrategia de aproximaciones para encontrar las regiones con similitud, logrando reducir considerablemente los tiempos de ejecución utilizando también un procesador, pero entregando resultados menos detallados que DOTTER con lo que se puede perder información de interés.

Procesar información a escala genómica usando técnicas seriales de computación (utilizando un solo procesador) generan retrasos en el análisis de datos bioinformáticos [15]. En contraste, los nuevos algoritmos que usan estrategias paralelas están en la capacidad de utilizar diferentes números de procesadores para realizar análisis independientes del tamaño de la información de entrada, mejorando así tiempos de ejecución y posibilitando el uso de servidores de cómputo para procesar toda la información disponible para cada análisis, lo cual abre nuevas oportunidades para los investigadores [16]. También se hace necesarias las técnicas novedosas de computación de alto rendimiento (HPC por sus siglas en inglés) para procesar esta información de manera precisa y eficiente [17], logrando resultados de buena calidad en tiempos muy cortos, facilitando así posteriores análisis y ajustes en las comparaciones incluso para genomas de gran tamaño. Existen aplicaciones que utilizan técnicas de HPC como HPC-BLAST [18] para alineamiento de secuencias, Inpactor [19] para identificar elementos repetitivos o HPC-CLUST [20] para agrupamiento de grandes secuencias de nucleótidos, estos son frecuentemente utilizados para ejecutar análisis bioinformáticos de forma eficiente y demuestran la utilidad de HPC en las diversas tareas bioinformáticas que se realizan.

Por otro lado, son pocas las aplicaciones que técnicas de HPC para realizar alineamiento gráfico de secuencias. Es por esto que en el presente artículo , propone una estrategia que permite analizar y procesar grandes cantidades de datos de forma rápida y eficiente utilizando lenguajes de programación de alto nivel como Python [21] y técnicas de HPC. Además, mejora los tiempos en los que se generan los resultados del alineamiento gráfico sin perder calidad con un algoritmo que trabaje sobre múltiples procesadores.

Este tipo de estrategias resultan de gran utilidad en las ciencias biológicas ya que permiten de manera visual determinar zonas repetitivas, reorganizaciones y mutaciones a nivel de cromosomas, además, facilitan la comparación de genomas entre especies diferentes para distinguir rasgos comunes [22]. En humanos puede contribuir a la rápida y correcta visualización e identificación de zonas repetitivas como los HERV (*Human Endogenous Retroviruses* por sus siglas en inglés) [23], debido a la estrecha relación que se ha demostrado de estos retrovirus ligados a enfermedades humanas como el cáncer de mama [24] o la leucemia [25].

II. Materiales y Métodos

II. a. Librerías: Para la implementación del algoritmo se utilizó Python versión 3 como lenguaje de programación, utilizando las librerías Numpy [26], Matplotlib [27], Sys, Time, Getopt y Multiprocessing, las cuales son esenciales para el manejo de arreglos, gráficos, lectura de archivos, toma de tiempos, lectura de parámetros por línea de comandos y trabajo sobre múltiples procesadores. La instalación de los paquetes se realizó en un ambiente de Anaconda. En la Tabla 1 se describen las versiones de cada uno de los softwares y librerías utilizadas en la investigación.

Tabla 1. Versiones de software y librerías

Paquete	***Versión***
Anaconda	4.6.14
Python	3.7.4
Numpy	1.17.2
Matplotlib	3.1.1
Sys	3.7
Time	3.7
Multiprocessing	3.7
Getopt	3.7

II. b. Algoritmo: El algoritmo de alineamiento gráfico implementado recibe como parámetros dos archivos a analizar en formato FASTA (los cuales pueden contener múltiples secuencias), el número de procesadores (CPUs) sobre los que se desea ejecutar el algoritmo y el número de bases a comparar por cada píxel llamada ventana; además se permiten modificar parámetros como el ancho, alto de la imagen. Inicialmente se unen todas las secuencias presentes en cada archivo para generar una única secuencia, luego se realiza el alineamiento dividiendo la secuencia en cadenas más pequeñas (ventanas) de tamaño definido por el usuario, de lo contrario el algoritmo lo asigna por defecto que de acuerdo con lo propuesto por DOTTER es una ventana de 25 bases, esta operación se realiza en simultaneo para la cantidad de procesadores definidos por el usuario. El algoritmo utiliza solo el sistema de puntuación de *match* y *mismatch* para comparar localmente las ventanas asignando un puntaje positivo cuando el par de bases son iguales (*match*) y negativo cuando el par de bases es negativo (*mismatch*) el algoritmo tiene definido el puntaje de *match* como +5 y el de *mismatch* como -4 como lo sugiere DOTTER. En el Pseudocódigo 1 describe cómo se realiza el cálculo del puntaje para dos cadenas:

Pseudocódigo 1. Función del cálculo del puntaje de alineamiento por *match* y *mismatch*.

```
Enteros:  N1 {Longitud de la secuencia1}
          N2 {Longitud de la secuencia2}
          match_puntaje=5
          mismatch_puntaje=-4
Vectores: V_mismatch = []
Function score (secuencia1, secuencia2)
si   N1 > N2 entonces
      longitud ← N1
sino
      longitud ← N2
fin_si
pos=0
     para i ← 0 hasta longitud; hacer
       si secuencia1[i]! = secuencia2[i]
       entonces
             v_mismatch[pos]← 1
             pos++
     fin_si
fin_para
num_mis ← sumatoria(v_mismatch) + absoluto(N1-N2)
mismatch_valor ← num_mis * mismatch_puntaje
     match_valor ←(longitud - sumatoria(v_mis
     match)) * match_puntaje
     puntaje_total ← mismatch_valor + match valor
Retornar puntaje_total
```

Luego de obtener los resultados de cada alineamiento y guardarlo en una matriz, se mapean los puntajes obtenidos a intensidades entre 0 y 255 siendo el valor más alto 255 y 0 el más bajo. Por último, se le aplican a la matriz de puntos filtros de reducción de ruido para mejorar la calidad y el detalle del alineamiento realizado para finalmente guardar la imagen en formato de gráficos de red portátiles (PNG por sus siglas en inglés).

II. c. Paralelización: El proceso de paralelización se realizó por medio de la librería Multiprocessing de Python. Para este proceso se tienen como entradas las dos secuencias a alinear gráficamente, las cuales se organizan en una matriz de tamaño NxM (donde N es la longitud de la secuencia uno y M es la longitud de la secuencia dos) en la que a una secuencia le corresponden los ejes horizontales y a la otra secuencia los ejes verticales de la matriz. Posteriormente, dependiendo del número de procesadores asignados por el usuario, se divide equitativamente la matriz inicial en n sub-matrices (donde n es el número de procesadores) y se almacenan los índices de la matriz inicial al que corresponde cada sub-matriz. Cada una de las sub-matrices realiza al mismo tiempo el cálculo de los puntajes del alineamiento y los puntajes son retornados a un proceso maestro que se encarga de unir las matrices de acuerdo con los índices previamente analizados y finalmente, generar la matriz de puntajes final de igual tamaño a la matriz original (Ver Fig. 1).

II. d. Datos de experimentación: Para ejecutar el las pruebas de rendimiento se utilizó como referencia un archivo FASTA del cromosoma 21 del Humano (*Homo sapiens*) [28] que contiene aproximadamente 46 millones de bases, disponible en la página del NCBI en: https://www.ncbi.nlm.nih.gov/nuccore/NC_000021.9.

Con esta información se ejecutó el algoritmo para comparar los tiempos de ejecución aplicando estrategias paralelas en comparación con los tiempos de ejecución obtenidos por GEPARD y de la misma manera para contrastar la eficiencia en términos de calidad del alineamiento gráfico.

II. e. Experimentos: Para el Cromosoma 21 del *Homo sapiens* se ejecutó el algoritmo en paralelo cinco veces para diferentes números de procesadores (1, 2, 4, 8, 12, 16, 20, 24, 34, 48 y 64) y se registraron los tiempos de ejecución en cada caso. Se compararon los tiempos obtenidos contra GEPARD para el mismo cromosoma. Para todos los experimentos se reservaron la misma cantidad de procesadores (64) para asegurar que no existieran otros procesos ejecutándose al tiempo en el nodo que pudieran afectar los tiempos de ejecución.

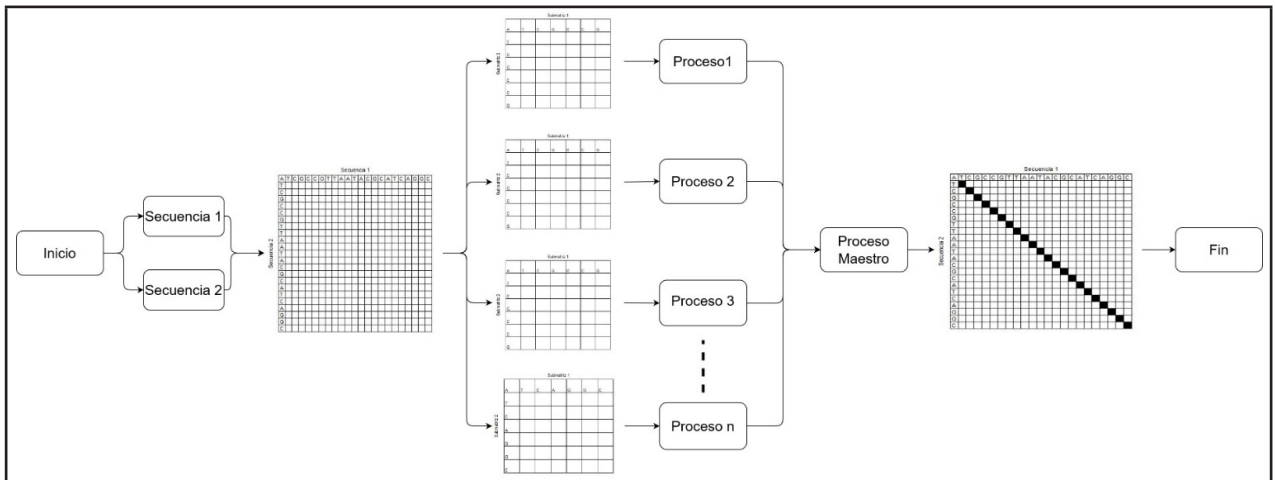

Fig. 1. Diagrama del proceso de paralelización del algoritmo de alineamiento gráfico.

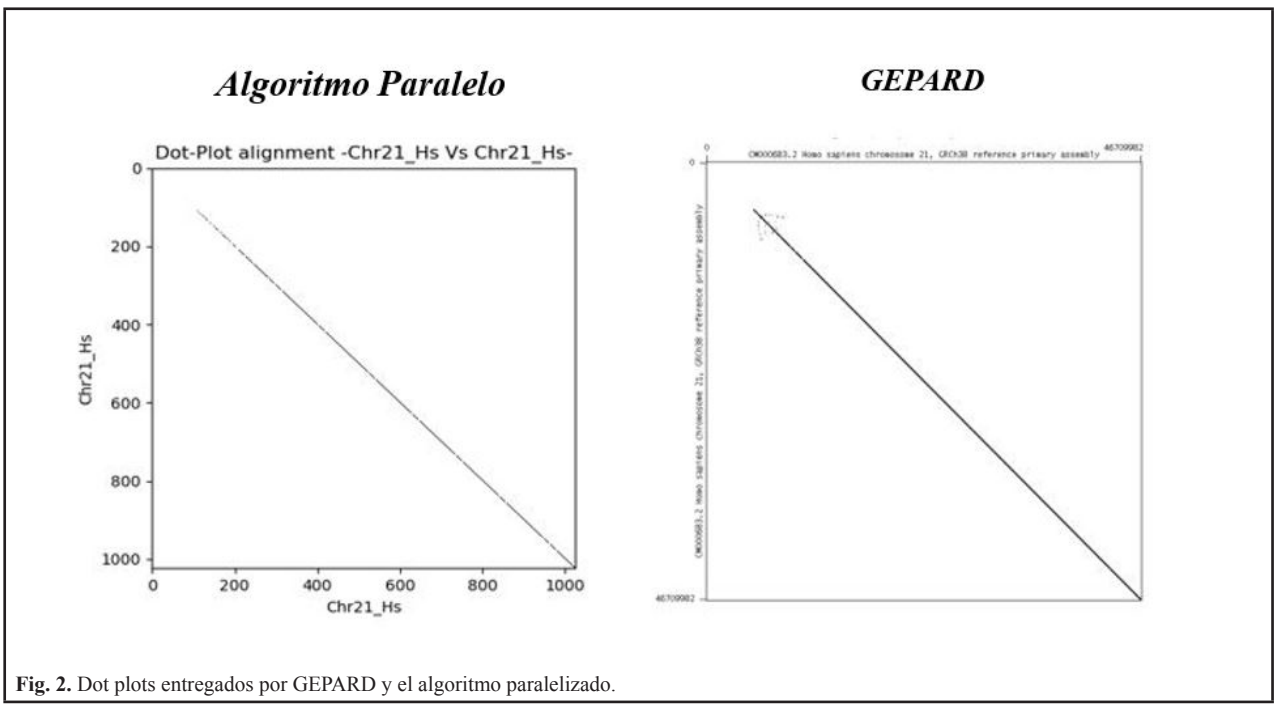

Fig. 2. Dot plots entregados por GEPARD y el algoritmo paralelizado.

II. f. Arquitectura computacional: Los experimentos se ejecutaron en un servidor con 64 procesadores Intel(R) Xeon(R) CPU E5-4650, 512 GB de memoria RAM y sistema operativo Ubuntu Server 18.04.3, gestionado por el calendarizador Slurm [29]. Todos los paquetes utilizados en este trabajo (ver Tabla 1) se cargaron directamente de los módulos provistos por el servidor.

II. g. Disponibilidad del algoritmo: La herramienta presentada en este trabajo es de acceso libre. El código fuente y la guía de uso e instalación están disponibles en: https://github.com/simonorozcoarias/parallelDotPlot.

III. Resultados

Los experimentos realizados durante esta investigación incluyen la implementación de un algoritmo usando estrategias paralelas. El algoritmo inicialmente se ejecutó con un procesador y tomando como entrada el cromosoma 21 del *Homo sapiens* para compararlo contra él mismo, se registró el tiempo de ejecución y el entregado por el software GEPARD (Ver Tabla 2). La Fig. 2 contiene el alineamiento gráfico generado por la estrategia propuesta y por GEPARD. Posteriormente se ejecutó el algoritmo incrementado el número de procesadores (2, 4, 8, 12, 16,

20, 24, 34, 48 y 64) analizando el cromosoma 21 contra él mismo y registrando los tiempos de ejecución (5 veces por cada número de procesadores). Finalmente Se calculó el tiempo promedio de ejecución, desviación estándar y la aceleración obtenida para cada número de procesadores; los resultados se encuentran en la Tabla 3.

Tabla 2. Tiempo de ejecución de GEPARD respecto al algoritmo.

Software	*Tiempo [S]*
GEPARD	4.762
Algoritmo en paralelo (64 CPUs)	238,6

De acuerdo con los datos anteriores registrados en la Tabla 3, en la Fig. 3 se muestra el gráfico de los tiempos promedios obtenidos para los diferentes números de CPUs.

En la Fig. 4 se muestran las aceleraciones del algoritmo que se obtuvieron al dividir el tiempo promedio para cada número de procesadores sobre el tiempo obtenido con un (1) procesador (tiempo nominal); Adicionalmente, en la Fig. 5 se presenta el comportamiento de los tiempos tomados para cada número de procesadores con un gráfico de caja y bigote en el que está contenida la desviación estándar de los tiempos de ejecución.

Tabla 3. Registro de tiempos del algoritmo para cada experimento y estadísticas.

	Tiempo por número de ejecución [S]								
CPUs	***1***	***2***	***3***	***4***	***5***	***Media [S]***	***tiempo [hh:mm:ss]***	***Desviación Estándar***	***Aceleración***
1	6.611	6.696	6.696	6.641	6.639	6.657	1:50:57	33,87	1,000
2	3.340	3.342	3.347	3.338	3.331	3.340	0:55:40	5,24	1,993
4	1.686	1.666	1.684	1.671	1.672	1.676	0:27:56	7,81	3,972
8	869	859	870	869	866	867	0:14:27	4,03	7,681
12	600	619	608	606	613	609	0:10:09	6,43	10,927
16	476	470	470	471	486	475	0:07:55	6,12	14,026
20	399	378	383	381	377	384	0:06:24	7,99	17,353
24	324	328	324	323	323	324	0:05:24	1,85	20,520
32	284	282	276	284	277	281	0:04:41	3,44	23,723
48	270	257	264	262	264	263	0:04:23	4,18	25,272
64	237	238	242	237	239	239	0:03:59	1,85	27,899

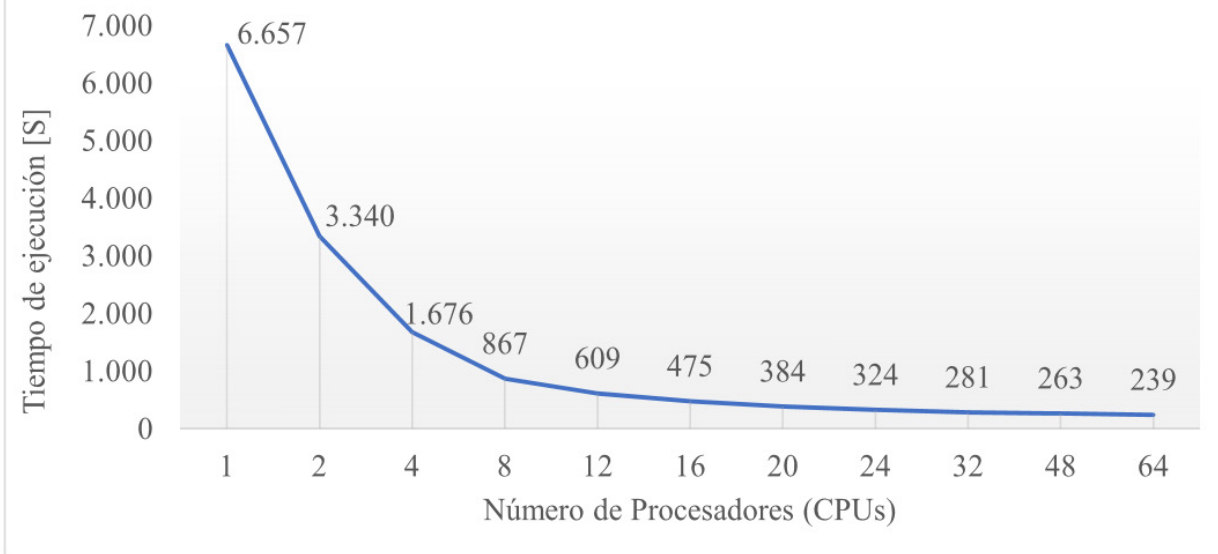

Fig. 3. Tiempo promedio de ejecución en función del número de CPUs.

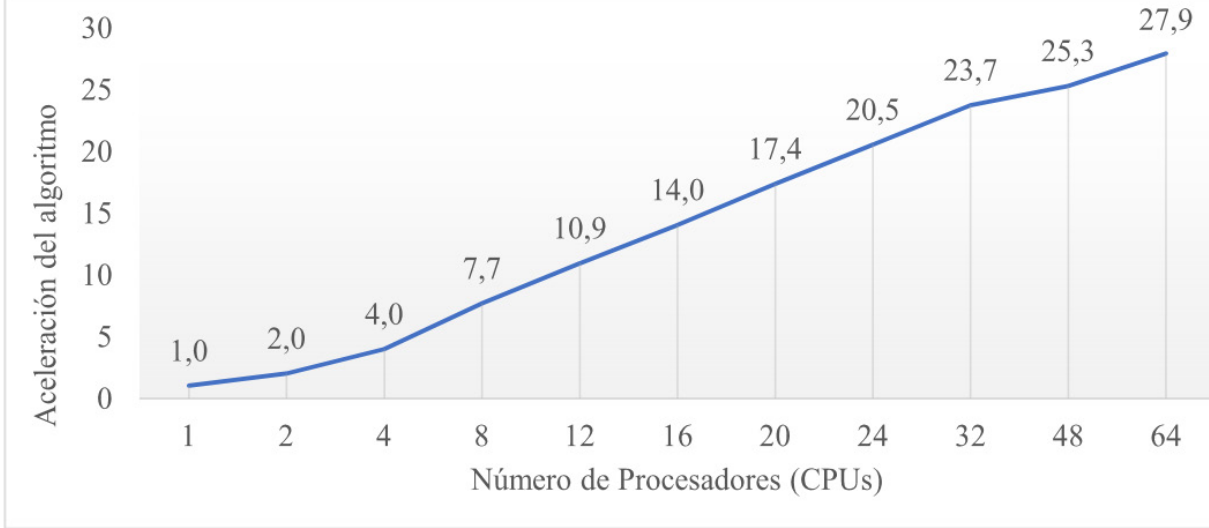

Fig. 4. Aceleración del algoritmo en función del número de CPUs.

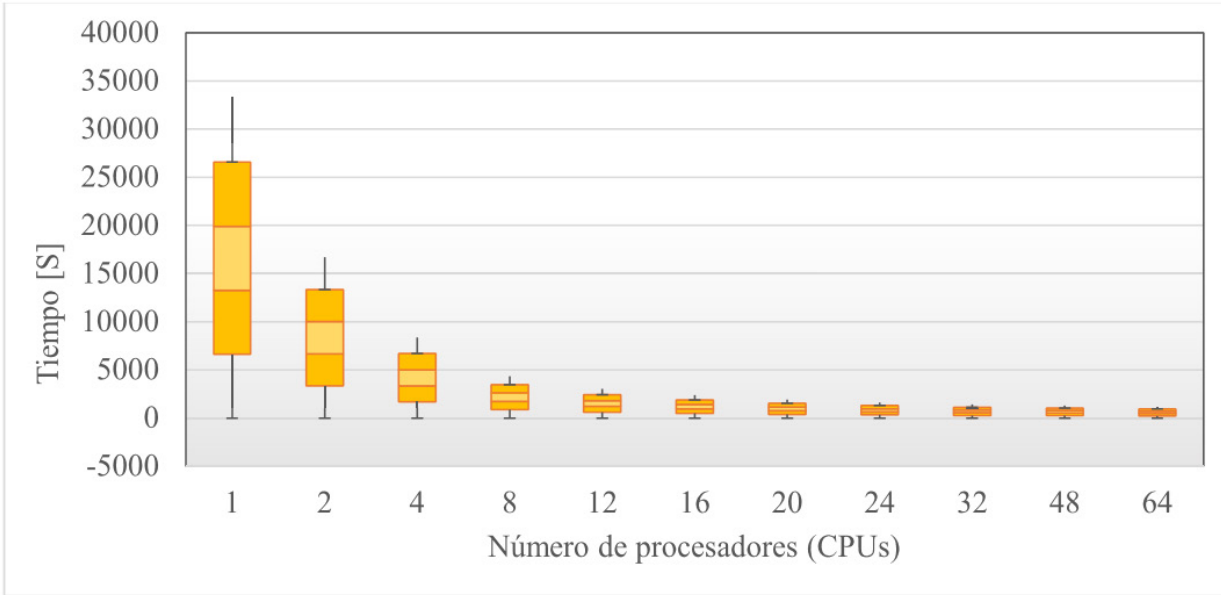

Fig. 5. Gráfico de los datos de los experimentos por cuartiles y su desviación estándar.

IV. Discusión de resultados

De acuerdo con la Fig. 3 Podemos observar que los tiempos de ejecución se reducen considerablemente en función del número de procesadores que se utilicen, así mismo, en la Fig. 4 se puede observar que la curva de aceleración del algoritmo crece de manera constante a medida que se aumentan el número de procesadores. Se obtuvieron aceleraciones de hasta 27,9 veces. La calidad del alineamiento gráfico obtenido respecto a Gepard es muy cercana utilizando el sistema de puntaje solo de *match* y *mismatch*. Adicionalmente, se observa que la curva de crecimiento de la aceleración (Fig. 4) no se estabiliza en ningún punto, evidenciando que si se usa mayor cantidad de CPUs (mayor a 64), la aceleración podría seguir creciendo hasta que se alcance un punto de saturación. Esto demuestra que la estrategia de paralelización utilizada en este trabajo es exitosa para el problema del alineamiento gráfico. En la Fig. 5 se puede observar que al ejecutar el programa con 2 o más procesadores se obtuvieron valores pequeños de las desviaciones estándar de los tiempos de ejecución de las cinco repeticiones del experimento por cada número de procesadores, esto permite deducir que el

algoritmo tiene una alta precisión debido a que los valores de tiempos son muy cercanos entre sí.

Utilizar la función para el cálculo del puntaje del alineamiento a través del *match* y *mismatch* permite reducir considerablemente la complejidad computacional tanto en tiempo de cómputo como en uso de memoria, debido a que el uso de algoritmos como Smith Waterman calcula el puntaje óptimo a través estrategias de programación dinámica. Estas estrategias obtienen mucha más información en la comparación de secuencias debido a que usan además de los *match* y *mismatch,* variables involucradas en los procesos de alineamiento como la penalidad de abertura (*Gap Open* en inglés), que es ampliamente utilizada en los alineamientos porque permite evaluar las posibles inserciones o deleciones que poseen las secuencias [30]. De esta forma se obtienen alineamientos mucho más precisos y los dot plots son de mejor calidad. Sin embargo, para el caso de los datos generados durante el periodo de la postgenómica, se prioriza el obtener resultados en tiempos prudentes (horas o minutos), que dot plots con un detalle muy alto, pero que se podrían demorar años (como en el caso de DOTTER). Por otro lado, al comparar secuencias a nivel genómico (cromosomas completos) se desea obtener información acerca de similitudes generales y no en detalle. Por esta razón, el uso de la función de *match* y *mismatch* podría ser adecuada para comparar de forma rápida las grandes cantidades de datos generados por investigaciones en áreas como la genómica o loa transcriptómica, sin embargo, los algoritmos tradicionales siguen siendo una mejor opción para encontrar similitudes detalladas en secuencias cortas (de unos pocos miles de bases).

V. Conclusiones

Los resultados obtenidos en esta investigación permiten concluir que es posible calcular el alineamiento gráfico dividiendo el dot plot en sub-matrices más pequeñas que son menos complejas de calcular para cada procesador y finalmente unir estos procesos para generar la matriz de puntajes correspondiente. Esta estrategia paralela es acertada y funcional ya que se logran aceleraciones de hasta 19,9 veces respecto a uno de los softwares reportados en la literatura como eficientes para este tipo de tareas (GEPARD). Además, se logró disminuir considerablemente los tiempos de ejecución (ver Fig. 3) de horas a unos pocos minutos (~4 min con 64 CPUs) logrando una aceleración de hasta 27,9 veces. Con los experimentos realizados se puede anotar que no se logró establecer la existencia de un punto de saturación o estabilización de la aceleración, con lo cual se plantea la posibilidad de que, para mayor número de procesadores, se pueden lograr mayores aceleraciones para el algoritmo hasta lograr un punto de saturación en un número mayor de procesadores que los utilizados en este trabajo y de esta manera disminuir considerablemente los tiempos de ejecución del algoritmo.

De acuerdo a lo anterior se plantea la importancia y utilidad de la estrategia paralelas en el campo de las ciencias computacionales y áreas de la bioinformática gracias al avance de los supercomputadores y al número de procesadores que estas puedan poseer, aumentar el número de CPUs se disminuye de manera exponencial el tiempo de ejecución hasta llegar a un punto de saturación, que no se logró determinar con los procesadores utilizados en las pruebas, contribuyendo con uno de los grandes retos de la era postgenómica en el análisis de la enorme cantidad de información disponible, lo que facilitaría, por ejemplo, estudios de genómica comparativa [8] para especies que comparten rasgos genéticos como el *Homo sapiens* y el *Pan troglodytes* [22]. También influyen en investigaciones de especies vegetales de gran tamaño como los del género *Pinus* (~20 GB - 40 GB) [31] que requieren recursos computacionales considerables en su análisis por el tamaño del genoma, además, ayudar en la eficiencia de las investigaciones realizadas reduciendo tiempos de espera de los resultados y posibilitando ajustes posteriores a los experimentos que se estén realizando.

Agradecimientos

Los autores reconocemos al IRD itrop HPC (South Green Platform) del IRD en Montpellier por proporcionar recursos de HPC que han contribuido a los resultados de la investigación reportados en este trabajo. URL: https://bioinfo.ird.fr/ - http://www.southgreen.fr. Reconocemos y agradecemos igualmente la colaboración de la Universidad Autónoma de Manizales por permitir el uso de sus instalaciones para el desarrollo de esta investigación, dentro del del proyecto 589-089.

Referencias

[1]. O. Lecompte, J. D. Thompson, F. Plewniak, J.-C. Thierry, and O. Poch, “Multiple alignment of complete sequences (MACS) in the post-genomic era,” *Gene*, vol. 270, no. 1, pp. 17–30, 2001, doi: https://doi.org/10.1016/S0378-1119(01)00461-9.

[2]. N. M. Luscombe, D. Greenbaum, and M. Gerstein, “A Proposed Definition and Overview of the Field,” *Methods Inf. Med.*, vol. 40, no. 4, pp. 346–358, 2001.

[3]. J. Arango-López, S. Orozco-Arias, J. A. Salazar, and R. Guyot, “Application of Data Mining Algorithms to Classify Biological Data: The Coffea canephora Genome Case,” in *Advances in Computing*, vol. 735, 2017, pp. 156–170.

[4]. S. P. Holmes and D. Gusfield, “Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology,” *J. Am. Stat. Assoc.*, vol. 94, no. 447, p. 989, 1999, doi: 10.2307/2670026.

[5]. T. Rognes and E. Seeberg, "Six-fold speed-up of Smith--Waterman sequence database searches using parallel processing on common microprocessors," *Bioinformatics*, vol. 16, no. 8, pp. 699–706, 2000.

[6]. B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biol.*, vol. 10, no. 3, p. R25, 2009.

[7]. S. Xiao, H. Lin, and W. Feng, "Accelerating protein sequence search in a heterogeneous computing system," in *2011 IEEE International Parallel & Distributed Processing Symposium*, 2011, pp. 1212–1222.

[8]. W. Chen, B. Liao, and W. Li, "Use of image texture analysis to find DNA sequence similarities," *J. Theor. Biol.*, vol. 455, pp. 1–6, 2018, doi: 10.1016/j.jtbi.2018.07.001.

[9]. B. Liao and T.-M. Wang, "New 2D graphical representation of DNA sequences," *J. Comput. Chem.*, vol. 25, no. 11, pp. 1364–1368, 2004, doi: 10.1002/jcc.20060.

[10]. T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, 1981.

[11]. S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970, doi: 10.1016/0022-2836(70)90057-4.

[12]. A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg, "Alignment of whole genomes," *Nucleic Acids Res.*, vol. 27, no. 11, pp. 2369–2376, 1999, doi: 10.1093/nar/27.11.2369.

[13]. E. L. L. Sonnhammer and R. Durbin, "A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis (Reprinted from Gene Combis, vol 167, pg GC1-GC10, 1996)," *Gene*, vol. 167, no. 1–2, pp. Gc1–Gc10, 1995, doi: 10.1016/0378-1119(95)00714-8.

[14]. J. Krumsiek, R. Arnold, and T. Rattei, "Gepard: A rapid and sensitive tool for creating dotplots on genome scale," *Bioinformatics*, vol. 23, no. 8, pp. 1026–1028, 2007, doi: 10.1093/bioinformatics/btm039.

[15]. B. Schmidt, *Bioinformatics: high performance parallel computer architectures*. CRC Press, 2010.

[16]. S. Orozco-Arias, R. Tabares-Soto, D. Ceballos, and R. Guyot, "Parallel Programming in Biological Sciences, Taking Advantage of Supercomputing in Genomics," in *Advances in Computing*, vol. 735, A. Solano and H. Ordoñez, Eds. Zurich: Springer, 2017, pp. 627–643.

[17]. D. Milone, A. Azar, and H. Rufiner, "Supercomputadoras basadas en 'clusters' de PCs," *Rev. Cienc.*, pp. 173–208, 2002.

[18]. S. Sawyer, M. Horton, C. Burdyshaw, and G. Brook, "HPC-BLAST : Distributed BLAST for Modern HPC Clusters," vol. 60, pp. 1–14, 2019.

[19]. S. Orozco-arias *et al.*, "Inpactor, Integrated and Parallel Analyzer and Classifier of LTR Retrotransposons and Its Application for Pineapple LTR Retrotransposons Diversity and Dynamics," *Biology (Basel).*, 2018, doi: 10.3390/biology7020032.

[20]. J. F. Matias Rodrigues and C. von Mering, "HPC-CLUST: distributed hierarchical clustering for large sets of nucleotide sequences," *Bioinformatics*, vol. 30, no. 2, pp. 287–288, 2014.

[21]. G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[22]. S. Schwartz *et al.*, "Human-mouse alignments with BLASTZ.," *Genome Res.*, vol. 13, no. 1, pp. 103–107, 2003, doi: 10.1101/gr.809403.

[23]. S. Hicks, D. A. Wheeler, S. E. Plon, and M. Kimmel, "Prediction of missense mutation functionality depends on both the algorithm and sequence alignment employed," *Hum. Mutat.*, vol. 32, no. 6, pp. 661–668, 2011, doi: 10.1002/humu.21490.

[24]. G. L. Johanning *et al.*, "Expression of human endogenous retrovirus-K is strongly associated with the basal-like breast cancer phenotype.," *Sci. Rep.*, vol. 7, no. 1, p. 41960, Dec. 2017, doi: 10.1038/srep41960.

[25]. S. Depil, C. Roche, P. Dussart, and L. Prin, "Expression of a human endogenous retrovirus, HERV-K, in the blood cells of leukemia patients," *Leukemia*, vol. 16, no. 2, pp. 254–259, 2002, doi: 10.1038/sj.leu.2402355.

[26]. S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.

[27]. J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[28]. M. Hattori *et al.*, "The DNA sequence of human chromosome 21 - supplement table," *Nature*, vol. 405, no. May, p. 7118, 2000.

[29]. M. Jette, A. Yoo, and M. Grondona, "SLURM: Simple linux utility for resource management," 2003, doi: 10.1007/10968987_3.

[30]. H. Carroll, P. Ridge, M. Clement, and Q. Snell, "Effects of gap open and gap extension penalties," *Proc. Third ...*, pp. 1–5, 2006, doi: 10.1.1.182.4493.

[31]. J. L. Wegrzyn *et al.*, "Unique features of the loblolly pine (Pinus taeda L.) megagenome revealed through sequence annotation," *Genetics*, vol. 196, no. 3, pp. 891–909, 2014, doi: 10.1534/genetics.113.159996.